

COTS-Based Systems: Lessons Learned from Experiences with COTS Software Use on Space Systems

10 September 2001

Prepared by

R. J. ADAMS and S. ESLINGER
Software Engineering Subdivision
Computer Systems Division

Prepared for

SPACE AND MISSILE SYSTEMS CENTER
AIR FORCE MATERIEL COMMAND
2430 E. El Segundo Boulevard
Los Angeles Air Force Base, CA 90245

Engineering and Technology Group


APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

20011017 018

This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-00-C-0009 with the Space and Missile Systems Center, 2430 E. El Segundo Blvd., Los Angeles Air Force Base, CA 90245. It was reviewed and approved for The Aerospace Corporation by M. A. Rich, Principal Director, Software Engineering Subdivision. Michael Zambrana was the project officer for the Mission-Oriented Investigation and Experimentation (MOIE) program.

This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.


Michael Zambrana
SMC/AXE

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.				
1. REPORT DATE (DD-MM-YYYY) 10-09-2001		2. REPORT TYPE		3. DATES COVERED (From - To)
4. TITLE AND SUBTITLE COTS-Based Systems: Lessons Learned from Experiences with COTS Software Use on Space Systems		5a. CONTRACT NUMBER F04701-00-C-0009		
		5b. GRANT NUMBER		
		5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Richard J. Adams and Suellen Eslinger		5d. PROJECT NUMBER		
		5e. TASK NUMBER		
		5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The Aerospace Corporation Laboratory Operations El Segundo, CA 90245-4691		8. PERFORMING ORGANIZATION REPORT NUMBER TR-2001(8550)-1		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space and Missile Systems Center Air Force Materiel Command 2430 E. El Segundo Blvd. Los Angeles Air Force Base, CA 90245		10. SPONSOR/MONITOR'S ACRONYM(S) SMC		
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) SMC-TR-01-19		
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				
13. SUPPLEMENTARY NOTES				
14. ABSTRACT The incorporation of COTS software into software-intensive systems brings promises of reduced cost and schedule, along with higher reliability and maintainability by using "proven" software. However, the reality of using COTS software can be very different! This report presents the results of a survey of USAF Space and Missile Systems Center (SMC) and National Reconnaissance Office (NRO) program experience with incorporating COTS software into their systems. Six major lessons learned derived from the survey are described, along with recommendations for improving the acquisition of COTS-based systems.				
15. SUBJECT TERMS Commercial off-the-shelf software, COTS software integration				
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 32
a. REPORT UNCLASSIFIED	b. ABSTRACT UNCLASSIFIED	c. THIS PAGE UNCLASSIFIED		
				19b. TELEPHONE NUMBER (include area code) Richard Adams (310)336-2907

Acknowledgements

The authors wish to thank the Aerospace Chief Architect/Engineer's Office for the opportunity to perform the COTS software study described in this report. The preparation of this report was funded by the Software Acquisition Mission-Oriented Investigative Experimentation (MOIE) task.

The authors also wish to thank the following members of the Computer Systems Division (CSD) and the Space-Based Surveillance Division (SBSD) for their review of this paper and suggestions for improvements.

Colleen Ellis, Senior Engineering Specialist, CSD

Sharon K. Hoting, Director, SBSD

Bonnie R. Troup, Manager, CSD

Contents

1.	Introduction.....	1
2.	COTS Software Study Process.....	3
	Lesson Learned #1.....	4
	Lesson Learned #2.....	5
	Lesson Learned #3.....	6
	Lesson Learned #4.....	7
	Lesson Learned #5.....	8
	Lesson Learned #6.....	9
3.	Acquisition Recommendations.....	13
4.	Conclusion	15
	References.....	17
	Appendix—Evaluation Criteria for COTS and Reuse Software	19

1. Introduction

The new DoD 5000 series of acquisition regulations¹⁻³ requires the exploitation of Commercial Off-the-Shelf (COTS) products in DoD acquisitions. The incorporation of COTS software into software-intensive systems brings promises of reduced cost and schedule, along with improved reliability and maintainability by using "proven" software. However, the reality is often very different! The use of COTS software, which can provide significant benefits, also poses major new risks in the acquisition, development, and sustainment of software-intensive systems, risks that must be acknowledged and managed.

In support of the USAF Space and Missile Systems Center's (SMC's) Directorate of Systems Acquisition, the authors performed an in-depth study of actual COTS-based system (CBS)* development and sustainment experiences on SMC and National Reconnaissance Office (NRO) programs. This study was motivated by numerous reports of COTS software-related problems. The purposes of this study were (1) to synthesize and share lessons learned from actual CBS development and sustainment experiences, and (2) to provide recommendations for mitigating the risks inherent in CBS development and sustainment.

* For this report, a COTS-based system (CBS) is defined to be a system that contains commercial-off-the-shelf *software* products as elements of the system.

2. COTS Software Study Process

The study process comprised three steps: gathering information, synthesizing lessons learned, and developing acquisition recommendations. The techniques selected for the first step were focused interviews and documentation reviews. A COTS software experience questionnaire was developed to provide a framework for the interviews. This questionnaire was sent to the interviewees in advance of the interview to enable them to prepare for the type of information being requested. The interview itself was then allowed to proceed as a free exchange of information, with the questionnaire being used to return conversation to the topic, if necessary. The questionnaire was also consulted toward the end of the interview to identify areas that had not been addressed. In addition, interviewees were asked to provide any written documentation previously prepared by their program on COTS software experiences or lessons learned.

The COTS software experience questionnaire requested information on both good and bad experiences with COTS software. For each experience, information was requested on the nature of the experience, where in the life cycle it occurred, the COTS software involved, the functions provided by the COTS software, and their criticality to the system. For good experiences, the interviewees were asked to identify any actions taken that contributed to the good experience. For bad experiences, the interviewees were asked to (1) identify any actions taken to solve the problem or mitigate further risk, and (2) provide information as to what they would have done differently to find the problem earlier and to solve the problem or mitigate the risk, given the benefit of perfect hindsight.

The authors interviewed over 50 representatives from 18 SMC and NRO program organizations, including personnel from the Government, the development contractors, and The Aerospace Corporation. In addition, domain experts from The Aerospace Corporation were interviewed concerning their experiences with COTS software in their domains. The domains chosen were considered to be critical to space systems, such as computer security and telemetry processing. Following the interviews, the authors reviewed the documentation on COTS software experiences and lessons learned obtained from the interviewees. Note that the information gathered was from experiences on the ground segments of space systems, due to the still rare use of COTS software in onboard software for satellites and launch vehicles.

During the second step of the study, the authors first identified and documented over 150 distinct findings from their interview notes and document reviews. The authors then performed an iterative series of analyses and syntheses to derive lessons learned from the findings. Six significant lessons learned were identified that encompassed the collection of findings. The final step of the study was to develop specific acquisition recommendations to help mitigate the risks in CBS development and sustainment. The lessons learned and recommendations are described below.

Lesson Learned #1

Critical aspects of CBS development and sustainment are out of the control of the customer, developer, and user.

This lesson concerns the realities of the commercial marketplace. COTS software vendors are driven by today's fast-paced market, characterized by highly volatile business strategies and market positions. Vendors may go out of business, or merge with or be acquired by other companies. Vendors may also drop or de-emphasize products or hardware platforms, usually without warning. This is especially true for high-performance UNIX platforms that are not the market leader for commercial industry but are used extensively in defense space applications.

Of particular note is the fact that the market for COTS software is driven by the more numerous commercial customers, not the defense community. In some instances, the market is diverging from defense needs. One example of this is the emphasis by some vendors on Windows NT platforms for commercial users (and the corresponding de-emphasis of the high-performance UNIX workstations used in ground systems for defense space applications). Another example is the targeting of COTS satellite control software to the operational paradigm of commercial communications satellite customers where there is minimal human intervention rather than the person-in-the-loop paradigm of defense satellite operations.

The quality and content of COTS software upgrades are unpredictable. Vendors are market driven in their upgrades, focusing upon additional features to attract new customers and fixes to problems encountered by their principal customer base. Vendors may not be willing to fix problems experienced by only a few customers, even if the customer is willing to pay the vendor to fix the product. This can be especially applicable to defense space applications, which may use different features or place different stresses upon the COTS software than commercial users of the same software.

Because of time-to-market pressure, vendors perform limited testing on COTS software upgrades, especially regression testing of supposedly unchanged features. In addition to introducing bugs in previously working capabilities, upgrades may decrease performance, increase computer resource utilization, and introduce incompatibilities with other COTS software products. Also, upgrades may eliminate backward compatibility with previous versions, possibly necessitating design and data structure rework. Numerous interviewees stressed the need for developers and sustainers to fully test each upgrade before incorporating it into the system.

The schedule of upgrades, both frequency and release dates, is time-to-market driven. However, the pressure to bring new features to market quickly may cause vendors to drop or slip other promised features, fixes, or upgrades for particular platforms. Sometimes needed upgrades are delayed because of dependencies between COTS software products (e.g., vendors waiting for the next operating system upgrade before issuing their next major upgrade).

The costs of COTS software products and associated services are also market driven. Fees and fee structures of licenses and services may change without warning, potentially resulting in a large cost impact if changes occur after developer commitment to a particular COTS software product. One

particularly damaging example of this is when a vendor eliminates site licenses and requires a separate copy of the COTS software to be purchased for each operator seat in the ground system. Vendors may also change the type and quality of the customer support they provide. In particular, new vendors trying to gain a market position may initially be very responsive but may lose their responsiveness after establishing a larger customer base.

No program organization interviewed had experienced all of the problems cited above, nor did any organization have problems with all of their selected COTS software. However, every program organization interviewed had some problems with one or more COTS software product. Encountering these realities of the commercial marketplace should be considered the norm, not the exception, in the development and sustainment of CBS. Contingencies (e.g., alternative product choices, cost and schedule margins) to handle these occurrences need to be built into development and sustainment plans from the beginning of the life cycle.

Lesson Learned #2

Full application of system and software engineering is required throughout the CBS life cycle.

This lesson reflects the fact that the use of COTS software does not eliminate portions of the system life cycle or the necessity for performing system and software engineering. The use of COTS software reduces, but does not eliminate, the scope of software design and implementation activities for that part of the software whose functionality is provided by the COTS software. Software requirements analysis, architectural design, integration and testing, and qualification testing must still be performed along with certain detailed design and implementation tasks. Moreover, system requirements analysis, design, integration and testing, and qualification testing must still be performed for all system functionality, independent of how the functionality is implemented.

Every new COTS software release requires a full application of the system and software engineering life cycle to properly incorporate the new release into the CBS. Incorporating each new release can require, for example, regression testing and prototyping to determine its behavioral characteristics as compared to the previous release and its compatibility with other COTS software in the CBS, testing of new features and bug fixes, installation and configuration of the COTS software in the CBS, modifications to glue code and user interfaces, modifications to the COTS software data base/file structure and content, full software and system integration testing and requirements verification, and training for both the software developers and the operators.

Thorough requirements analysis is especially important with CBS development since it is necessary to understand which requirements can be traded against existing COTS software capabilities versus which requirements are essential to the mission and not in the trade space. This is true for all levels of requirements from the highest level system requirements through the software level requirements. To understand existing COTS software capabilities, hands-on prototyping of the COTS software is necessary since it is not generally possible to determine the true COTS software capabilities from the vendor's marketing demonstrations and literature. Furthermore, due to the possibility of incompatibilities or adverse interactions (e.g., performance degradation) between COTS software products, this

prototyping must be performed in a system context where unexpected impacts of integrating multiple COTS software products can be discovered.

Numerous interviewees emphasized the importance of designing CBS architectures to support the evolution or replacement of COTS software. Since true "plug and play" among COTS software products does not yet exist in the commercial marketplace, architectural features that help minimize the impact of upgrading to new releases of COTS software or replacing one COTS software product with another of similar functionality are essential. The CBS architecture must also have a sufficient computer resource margin and growth path to accommodate increases in resource utilization by COTS software upgrades.

Security, safety, and supportability must be designed into the CBS at the system level. COTS software capabilities in these areas are aimed at commercial applications having different, and frequently less stringent, security, safety, and supportability requirements than defense applications. Furthermore, each COTS software product is designed independently, as a stand-alone package, not as part of an integrated system. The CBS design needs to provide for integrated security, safety, and supportability features across COTS software products and newly developed or reused code. This is especially important for security since each COTS software product has its own vulnerabilities. Many of these vulnerabilities are well known in the industry, and new vulnerabilities are continually being identified. Without integrated security features being designed into the CBS, the system's vulnerabilities can be determined simply by knowing which COTS software products are in use.

Both initial evaluation of COTS software for product selection and subsequent periodic re-evaluation of new COTS software releases for product evolution are necessary throughout the development and sustainment life cycle. The system engineering viewpoint must be applied simultaneously to the selection of the computer hardware and COTS software. Selection of a computer hardware platform without concurrent consideration of the availability of COTS software for that platform can result in more newly developed software being required than expected, and thus can increase the system development and life-cycle costs. The initial evaluations and periodic re-evaluations of COTS software must be based upon multi-dimensional evaluation criteria, not just upon the functionality provided by the COTS software. Examples of such evaluation criteria include the reliability of the COTS software, its ability to interface with other parts of the CBS and with legacy systems, the COTS software's implied operations concept, the vendor's characteristics, and the cost. A more complete list of key evaluation criteria is given in the Appendix. Finally, it is always necessary to have backup strategies and contingency plans for each COTS software product in case unforeseen problems arise that require its replacement.

Lesson Learned #3

CBS development and sustainment require a close, continuous, and active partnership among the customer, developer, and user.

This lesson concerns the need for the customer, developer, and user to be prepared to trade cost, schedule, performance, and operations and maintenance concepts to achieve the maximum benefits from using COTS software. The customer and user must understand their requirements sufficiently well to know which requirements can be relaxed to achieve a COTS-based solution and which are

essential to the mission and cannot be traded. To facilitate the trades of requirements versus COTS software capabilities, the customer and user must be willing to prioritize their requirements initially and re-prioritize them as necessary throughout the life cycle. Merging of an intimate understanding of the requirements (as held by the customer and user) and an intimate knowledge of the COTS software capabilities (as held by the developer) is necessary to ensure the adequacy of these trade decisions.

Each COTS software product has its own world view that, when incorporated into the CBS, may force a particular operations concept upon the user. Frequently, the COTS software operational paradigm is at odds with the user's existing operational procedures. As such, accommodating a COTS-based solution may require the user to be able and willing to re-engineer existing operational procedures. Similar re-engineering may be needed for existing on-site maintenance procedures. Ensuring the eventual acceptability of the CBS in the user's operational environment requires close cooperation between the user and developer.

At any time during the CBS life cycle, decisions may need to be made due to, for example, new COTS software limitations or incompatibilities being discovered, COTS software upgrades diverging from needs, or COTS software needing to be replaced due to withdrawal of vendor support. A close, continuous and active partnership among the customer, developer, and user (e.g., via the application of Integrated Product and Process Development) will help to ensure the adequacy of the major COTS software-related trade decisions and the acceptability of the delivered CBS. Without such a partnership, the customer and user will not gain a full understanding of the evolving CBS capabilities as system development proceeds and may experience unpleasant surprises when finally exposed to the capabilities of the delivered CBS in the operational environment.

Lesson Learned #4

Every CBS requires continuous evolution throughout development and sustainment.

This lesson reflects the fact that maintaining currency with COTS software upgrades is essential during both development and sustainment. Because vendors support only a limited number of past releases, delaying implementation of upgrades can result in unsupported versions of COTS software products in the CBS. When this happens, the vendor will not provide fixes to bugs and will not provide consultation services.

Delaying the implementation of upgrades can exacerbate system impacts. Upgrading from one major release to the next consecutive release does require time and effort. However, the upgrade can be considerably more expensive when attempting to skip major releases, which generally occur every 12 to 18 months. Delaying upgrades longer than that time interval can result in significant paradigm changes in the COTS software. Sometimes upgrading to a later major release requires upgrading through each of the intermediate major releases. This is especially true if the vendor has changed the structure of the COTS software's databases or files. When this occurs, vendors frequently provide automated tools to assist their customers in conversion from one release to the next consecutive release. Such tools are not provided to assist in skipping major releases.

Many factors, both internal and external to the CBS, can drive the need to maintain currency with upgrades to the CBS' COTS software. Organizations or systems external to the CBS can require COTS software upgrades. Examples of this include upgrades to Government off-the-shelf software incorporated into the CBS, to legacy systems with which the CBS must interface, or to the Defense Information Infrastructure Common Operating Environment (if used by the CBS). Another factor influencing the need to maintain currency with COTS software upgrades is the limited life span of computer hardware platforms (e.g., workstations, and servers). Most programs plan on hardware upgrades every 4 to 5 years during sustainment. Maintaining currency with COTS software releases is essential for upgrading to new hardware since it is not usually possible or desirable to execute old versions of the operating system and other COTS software on new hardware platforms. Furthermore, COTS software may need to be replaced or added at any time due to such factors as elimination of vendor support, divergence from system needs, identification of unacceptable limitations or vulnerabilities, increased costs for licenses or support services, and new or modified user needs requiring changes in functionality or performance. Incorporating new COTS software usually requires the latest version of the operating system and other related COTS software to be in place.

One of the most damaging decisions frequently made in CBS development is to freeze the versions of the COTS software products throughout the development period. Due to the length of the development period for large software-intensive defense systems, this decision can result in the delivery of a system that is obsolete because its COTS software products are no longer supported. A major upgrade effort with associated cost and schedule impacts is then necessary before or shortly after the system becomes operational. Maintaining currency with COTS software upgrades is necessary throughout development as well as sustainment. Upgrading COTS software needs to be built into both development and sustainment plans from the beginning of the life cycle.

Numerous interviewees emphasized the folly of modifying COTS software, which can constrain the CBS evolution path and increase life-cycle costs. Modifying COTS software should always be a solution of last resort in CBS design. Incorporating a modified COTS software product into the CBS requires the developer and Government to engage in a long-term relationship with the vendor to ensure that the unique modifications will be made to future releases. Attaining such a relationship is not always possible.

Lesson Learned #5

Current processes must be adapted for CBS acquisition, development, and sustainment.

This lesson concerns the need to modify existing processes to be suitable for the acquisition, development and sustainment of CBS. The developer's software and system engineering processes must be adapted to handle the integration of COTS software into the system. New processes must be added, and existing processes updated to handle such activities as performing requirements trades against COTS software capabilities; evaluating COTS software against robust evaluation criteria; accounting for COTS software in safety, security, and supportability analysis and design; and incorporating COTS software upgrades during development.

CBS development works best when iterative life-cycle models (e.g., spiral or evolutionary) and extensive prototyping of the COTS software in the system context are used together, and when the

understanding gained from COTS software prototyping is integrated with the software architecture and design models. Furthermore, the time and effort distribution for development tasks need to be reallocated. Additional time and effort need to be spent on evaluation, prototyping, and analysis (the front end), and on integration and testing (the back end), and less time and effort need to be spent on software implementation (the middle).

Numerous interviewees stressed the need for enhanced configuration management processes to handle the complexities of COTS software during both development and sustainment. The configuration management system must be able to manage multiple releases and patches to each release for each COTS software product. It must also be able to manage different configurations of COTS software at each development, sustainment, and operations facility (including mobile units), and even different configurations of COTS software on each computer hardware platform within each facility. For COTS software incorporated into firmware, configuration management cannot be performed at the board level, but must be performed for the contents of the chips on the board.

Customer and user processes also need to be created or adapted to be suitable for the acquisition and sustainment of CBS. Examples include prioritizing user requirements, providing flexible and efficient responses to unexpected impacts due to problems encountered with COTS software, and handling the schedule variability of COTS software upgrades. Other examples include developing contracts compatible with the acquisition of CBS and ensuring that program milestones are compatible with the reallocation of time needed for CBS development schedules.

Interviewees also stressed the need for standardization of certain Government processes as they relate to COTS software. Areas needing standardization include safety certification and security accreditation so that all parties understand the safety and security requirements that must be fulfilled when the system contains COTS software. In addition, standardized Government processes for COTS software licenses need to be implemented to ensure that COTS software license currency is maintained and that the COTS software licenses agreed to by the Government are suitable for defense needs. The license for a COTS software product to be used by operational forces in the field, for example, should prohibit expiring keys in the COTS software and should not contain any export restrictions.

Lesson Learned #6

Actual cost and schedule savings with CBS development and sustainment are overstated.

This lesson concerns the universal tendency to underestimate the required cost and schedule for CBS development and sustainment. There are two principal components to this underestimation: (1) completely overlooking or significantly underestimating tasks that must be performed in CBS development and sustainment or costs of COTS software license fees and other services, and (2) not allowing enough cost and schedule margin to handle unexpected impacts that can occur due to problems with COTS software at any time during the life cycle.

Examples of frequently overlooked tasks include hands-on prototyping of COTS software, especially in a system context; acquisition of in-depth knowledge of COTS software (e.g., training mentors and toolsmiths and purchasing vendor support); installation and configuration of the COTS software in

the development and operational facilities; and preparing integrated system training and documentation in addition to the vendor-supplied training and documentation. Also, tasks to incorporate COTS software upgrades are almost always overlooked. Examples of such tasks include performing COTS software and system regression tests for each COTS software upgrade; implementing and testing software changes needed to support the upgrades (e.g., additions or changes to glue code, databases, or configuration files); and training developers and operators for each COTS software upgrade. The time and effort for these overlooked tasks generally cannot be obtained from software cost models, but must be estimated bottom-up and included in the total cost and schedule estimates.

One area where time and effort are significantly underestimated is software engineering. Software development time and effort are generally obtained by estimating the number of source lines of code and applying a software cost model. When the decision is made to use COTS software to obtain certain system functionality, the total number of lines of code is reduced by the number of lines of code that would have been needed to provide that functionality. Using this technique with a software cost model causes the elimination of all software development activities for that functionality from the cost and schedule estimates. However, use of COTS software to provide functionality reduces only the amount of software design and implementation effort, not all software development activities. Software requirements analysis, architectural design, integration and testing, and qualification testing must still be performed, along with certain detailed design and implementation tasks. Even if the number of lines of glue code for integrating the COTS software is added to the total software size estimate, the resulting cost and schedule estimates are not sufficient to cover all of the necessary software development activities. While some of the newer software cost models do have features available for estimating costs associated with COTS software, these models are not yet in widespread use, and the accuracy of the resulting estimates has not yet been calibrated in the defense software environment.

Other areas where time and effort are significantly underestimated are system engineering and system integration and testing. The system models and tools currently in use for cost and schedule estimation do not adequately address the incorporation of COTS software. The effort for system engineering and system integration and testing is commonly estimated as a percentage of the total development cost. When COTS software is used to provide some system functionality, the reduction in the software development effort causes a corresponding reduction in the system engineering and system integration and test effort. This reduction is not warranted since the same system engineering and system integration and testing for that functionality must still be performed, independent of whether COTS software or developed code provides the functionality.

Costs of COTS software license fees are also frequently overlooked or underestimated. The number of different COTS software products required to implement the CBS and the number of individual licenses required to be purchased are difficult to estimate, especially early in the life cycle before the design is known. In addition, vendors usually charge for services not included in their standard licenses. Examples of such services are on-site vendor assistance during development or operations and escrowing source code to protect against the possibility of the vendor going out of business.

CBS cost and schedule estimates almost never contain enough margin to handle the COTS software problems encountered in CBS development and sustainment. As described above, unexpected impacts can occur with COTS software at any time during the life cycle. The lack of appropriate

margin results in cost and schedule overruns when COTS software-related problems occur. When Cost As An Independent Variable (CAIV) is applied, the cost of handling unexpected COTS software problems can mean that system capabilities must be deleted to balance the cost. Cost and schedule estimates for CBS development and sustainment should always contain a planned margin (i.e., management reserve) for handling the unexpected COTS software problems that are certain to arise.

3. Acquisition Recommendations

The Government needs to be an intelligent CBS buyer. Accomplishing this requires appropriate planning and contracting for CBS acquisition in addition to adapting the Government processes to support CBS development and sustainment (as described in lesson #3 above). Some of the important items for which the Government must plan and contract are cost and schedule management reserves for handling the unexpected COTS software-related problems; a COTS upgrade strategy for both development and sustainment; a close, continuous and active partnership among the customer, developer, and user; full life-cycle system and software engineering; and additional emphasis on safety, security, and supportability.

In particular, it should be noted that defense CBSs are almost never commercial items in themselves, but are large, complex, software-intensive systems, some of whose components contain COTS software. What is desired is a balanced solution among COTS, reuse, and newly developed software to meet the CBS cost, schedule, and performance objectives. Therefore, commercial item procurements (i.e., FAR 12 acquisitions) are almost never appropriate vehicles for acquiring defense CBSs.

To support defense programs in acquiring CBSs, it is recommended that several cross-program horizontal engineering initiatives be established. First, guidance for CBS life-cycle cost and schedule estimation needs to be developed to address the problems described in lesson #6. Second, a repository for actual development and sustainment experiences with COTS software products (as opposed to vendor marketing information) needs to be developed and made accessible to CBS acquirers, developers, and sustainers. Finally, specific CBS acquisition guidance that can be tailored to individual programs is needed, such as recommended contract structures, language for incorporation into contracts, and guidance for applying evolutionary acquisition.

4. Conclusion

The potential benefits of using COTS software in defense systems are extensive. Today's complex defense systems require the leverage provided by COTS software, that is, enhanced system capabilities with reduced cost and schedule. The use of COTS software enables the Government and developers to focus on providing the defense-unique needs.

This study demonstrated, however, that only careful acquisition, development, and sustainment preparation and execution achieve the potential CBS benefits. CBS success depends upon preparing for a complex development and sustainment effort; preparing for inherent cost, schedule, and performance risks beyond Government or developer control; and preparing to make adjustments to current acquisition, development, and sustainment processes. While this study was conducted on defense space systems, the authors believe that the lessons learned are not limited to that domain, but are widely applicable to the use of COTS software in any large, software-intensive system.

References

1. DoD 5000.2-R (Interim), Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs, 10 June 2001.
2. DoDD 5000.1, The Defense Acquisition System, (Incorporating Change 1, January 4, 2001), 23 October 2000.
3. DoDI 5000.2, Operation of the Defense Acquisition System, (Including Change 1), 4 January 2001.
4. Eslinger, S., "Software Acquisition and Software Engineering Best Practices," Aerospace Technical Report TR-2000(8550)-1, The Aerospace Corporation, 15 November 1999.
5. IEEE/EIA Interim Standard J-STD-016-1995, Standard for Information Technology, Software Life Cycle Processes, Software Development Acquirer-Supplier Agreement, 30 September 1995.

Appendix—Evaluation Criteria for COTS and Reuse Software

One of the significant problems in the use of COTS and reuse software is the lack of a thorough evaluation before the decision is made to incorporate the COTS and reuse software into the system under development. Such an evaluation should be the basis of any decision to use, or not use, particular COTS software packages or reuse software. If, after the evaluation, the decision is made to use the COTS or reuse software, the results of the evaluation will identify specific risks in incorporating that COTS or reuse software into the deliverable software product. Thus, the evaluation will assist in determining risk mitigation efforts that should be carried out. In addition, after the decision is made to use the COTS or reuse software, that software must be the subject of continuous risk management throughout the development life cycle. This includes frequent re-evaluation of the COTS and reuse software as the COTS/reuse software evolves and the system and software development progresses.

Essential criteria recommended by the authors for evaluating COTS and reuse software are shown in Table A-1.[†]

Table A-1. Recommended Criteria for Evaluating COTS and Reuse Software Products

Ability to provide required capabilities and meet required constraints

Ability to satisfy requirements

Ability to achieve necessary performance, especially with realistic operational workloads

Appropriateness of algorithms in the COTS/reuse software for use in the new system

As evidenced by characterization/stress testing within the system context to determine capabilities and performance

Ability to provide required protection (safety, security, and privacy)

Provided inherently in the COTS or reuse software product, or

Able to be provided around the COTS/reuse software product by system design features

Reliability/maturity

As evidenced by an established track record

As evidenced by prototype evaluation within the system context

Testability

As evidenced by the ability to identify and isolate faults

Operability

Suitability of the COTS/reuse software's implied operations concept to the operations concept of the new system

[†] Starting with the list of evaluation criteria for reusable software products provided in IEEE/EIA J-STD-016-1995,⁵ the authors added to that list over several years, based on their experiences with COTS and reuse software on numerous programs. The authors' set of evaluation criteria was first published in 1999 in the Aerospace Technical Report "Software Acquisition and Software Engineering Best Practices."⁴ The authors have significantly enhanced their previously published set of evaluation criteria based upon the results of this study.

Interoperability with other system and system-external elements

- Compatibility with system interfaces
- Adherence to standards (e.g., open systems interface standards)
- Ability to interface with legacy systems

Suitability for incorporation into the new system architecture

- Compatible software architecture and design features
- Absence of obsolete technologies
- Need for re-engineering and/or additional code development (e.g., wraps, "glue" code)
- Compatibility among the set of COTS software packages
- As evidenced by prototyping within the system context (e.g., to determine compatibility, wraps, "glue" code)

Ability to remove or disable features/capabilities not required in the new system

- Impact if those features cannot be removed/disabled or are not removed/disabled
- As evidenced by prototyping within the system context

COTS or reuse software supplier viability

- Compatibility of COTS or reuse supplier's future direction with program needs (including both software and platform emphasis)
- Supplier long-term commitment to COTS or reuse software product
- Supplier long-term business prospects
- Type of supplier support available
- Quality of supplier support available

Availability of personnel knowledgeable about the COTS/reuse product

- Training required
- Hiring required
- Vendor or third-party support required

Availability and quality of documentation and source files

- Completeness
- Accuracy

Acceptability of software product licensing and data rights

- Restrictions on copying/distributing the software or documentation
- License or other fees applicable to each copy
- Acquirer's usage and ownership rights, especially to the source code
 - Ability to place source code in escrow against the possibility of the vendor/developer going out of business
- Warranties available
- Absence of unacceptable restrictions in standard license, e.g.
 - Export restrictions
 - Expiring keys

Supportability

- Suitability of the COTS/reuse software product's support paradigm (e.g., distribution, installation) to the support concept of the new system, especially for mobile or remote sites

Maintainability, including:

- Likelihood the software product will need to be changed
- Feasibility/difficulty of accomplishing that change if changes are to be made by the program reusing the software product
 - Quality of design, code and documentation

- Need for re-engineering and/or restructuring
- Feasibility/difficulty of accomplishing that change, if changes are to be made by the vendor or product developer (e.g., for COTS or proprietary software)
- Priority of changes required by this program versus other changes being made
- Likelihood that the current version will continue to be maintained by the vendor/developer
- Likelihood of being able to modify future versions to include changes
- Impact on life cycle cost
- Impact if the current version is not maintained by the vendor/developer or changes are not able to be incorporated into future versions

Impacts of upgrades to COTS or reuse software products

- Frequency of COTS/reuse upgrades/modifications being made by the vendor/developer (i.e., of a new version being released) after a particular version has been incorporated into the system
- Feasibility/difficulty of incorporating the new version of the COTS/reuse product into the system
- Impact if the new version is not incorporated
- Ability of the system/software architecture (of the new system) to support the evolution of COTS/reuse software products

Compatibility of planned upgrades of COTS or reuse software with software development plans and schedules

- Compatibility of planned upgrades with build content and schedules
- Impact on development cost and schedule to incorporate upgrades
- Dependencies among COTS and reuse software products
 - Potential for an incompatible set of COTS and/or reuse software products
 - Potential for schedule delays until all dependent COTS and reuse software products are upgraded

Criticality of the functionality provided by the COTS or reuse software

- Availability of alternate source(s) for the functionality

Short- and long-term cost impacts of using the COTS/reuse software

- Amount of management reserve needed to handle uncertainties
 - For example, less COTS/reuse software usable; more newly developed software required; COTS/reuse limitations identified

Technical, cost, and schedule risks and tradeoffs in using the COTS or reuse software product

- Ability to tolerate COTS or reuse software problems beyond the program's control at any point in the system life cycle
- Ability to incorporate continuous evolution of COTS or reuse products during development and sustainment

LABORATORY OPERATIONS

The Aerospace Corporation functions as an "architect-engineer" for national security programs, specializing in advanced military space systems. The Corporation's Laboratory Operations supports the effective and timely development and operation of national security systems through scientific research and the application of advanced technology. Vital to the success of the Corporation is the technical staff's wide-ranging expertise and its ability to stay abreast of new technological developments and program support issues associated with rapidly evolving space systems. Contributing capabilities are provided by these individual organizations:

Electronics and Photonics Laboratory: Microelectronics, VLSI reliability, failure analysis, solid-state device physics, compound semiconductors, radiation effects, infrared and CCD detector devices, data storage and display technologies; lasers and electro-optics, solid state laser design, micro-optics, optical communications, and fiber optic sensors; atomic frequency standards, applied laser spectroscopy, laser chemistry, atmospheric propagation and beam control, LIDAR/LADAR remote sensing; solar cell and array testing and evaluation, battery electrochemistry, battery testing and evaluation.

Space Materials Laboratory: Evaluation and characterizations of new materials and processing techniques: metals, alloys, ceramics, polymers, thin films, and composites; development of advanced deposition processes; nondestructive evaluation, component failure analysis and reliability; structural mechanics, fracture mechanics, and stress corrosion; analysis and evaluation of materials at cryogenic and elevated temperatures; launch vehicle fluid mechanics, heat transfer and flight dynamics; aerothermodynamics; chemical and electric propulsion; environmental chemistry; combustion processes; space environment effects on materials, hardening and vulnerability assessment; contamination, thermal and structural control; lubrication and surface phenomena.

Space Science Applications Laboratory: Magnetospheric, auroral and cosmic ray physics, wave-particle interactions, magnetospheric plasma waves; atmospheric and ionospheric physics, density and composition of the upper atmosphere, remote sensing using atmospheric radiation; solar physics, infrared astronomy, infrared signature analysis; infrared surveillance, imaging, remote sensing, and hyperspectral imaging; effects of solar activity, magnetic storms and nuclear explosions on the Earth's atmosphere, ionosphere and magnetosphere; effects of electromagnetic and particulate radiations on space systems; space instrumentation, design fabrication and test; environmental chemistry, trace detection; atmospheric chemical reactions, atmospheric optics, light scattering, state-specific chemical reactions and radiative signatures of missile plumes.

Center for Microtechnology: Microelectromechanical systems (MEMS) for space applications; assessment of microtechnology space applications; laser micromachining; laser-surface physical and chemical interactions; micropropulsion; micro- and nanosatellite mission analysis; intelligent microinstruments for monitoring space and launch system environments.

Office of Spectral Applications: Multispectral and hyperspectral sensor development; data analysis and algorithm development; applications of multispectral and hyperspectral imagery to defense, civil space, commercial, and environmental missions.



**THE AEROSPACE
CORPORATION**

2350 E. El Segundo Boulevard
El Segundo, California 90245-4691
U.S.A.